Solution 1: AdaBoost - Updates

(a) For the first case, only one sample is incorrect. The weighted in-sample misclassification rate can be calculated as:

$$\text{err}^{[m]} = \sum_{i=1}^{n} w^{[m](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}} = 0.01 \tag{1}$$

We can now calculate the weight of the base learner:

$$\hat{\beta}^{[m]} = \frac{1}{2} \log\left(\frac{1 - \text{err}^{[m]}}{\text{err}^{[m]}}\right) = 0.5 \cdot log\left(\frac{0.99}{0.01}\right) = log(\sqrt{99}) \tag{2}$$

The updated weights are then calculated as:

$$\begin{aligned} w^{[m+1](i)} &= w^{[m](i)} \cdot \exp\left(-\hat{\beta}^{[m]} \cdot \underbrace{y^{(i)} \cdot \hat{h}(\mathbf{x}^{(i)})}_{\{-1,1\}}\right) \\ &= \begin{cases} 0.01 \cdot \exp\left(-log(\sqrt{99}) \cdot 1 \cdot 1\right) & \text{if } i = 1, \ldots, 4, 6, \ldots, 10 \\ 0.01 \cdot \exp\left(-log(\sqrt{99}) \cdot 1 \cdot -1\right) & \text{if } i = 5 \\ 0.1 \cdot \exp\left(-log(\sqrt{99}) \cdot 1 \cdot 1\right) & \text{if } i = 11, \ldots, 19 \end{cases} \\ &= \begin{cases} 0.001 & \text{if } i = 1, \ldots, 4, 6, \ldots, 10 \\ 0.0995 & \text{if } i = 5 \\ 0.01 & \text{if } i = 11, \ldots, 19 \end{cases} \end{aligned} \tag{3}$$

(b) In the second case, five out of the 9 samples with weight 0.1 are misclassified. The weighted in-sample misclassification rate can be calculated as:

$$\text{err}^{[m]} = \sum_{i=1}^{n} w^{[m](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}} = 0.5 \tag{4}$$

We can now calculate the weight of the base learner:

$$\hat{\beta}^{[m]} = \frac{1}{2} \log\left(\frac{1 - \text{err}^{[m]}}{\text{err}^{[m]}}\right) = 0.5 \cdot log\left(\frac{0.5}{0.5}\right) = 0 \tag{5}$$

The updated weights are then calculated as:

$$\begin{aligned} w^{[m+1](i)} &= w^{[m](i)} \cdot \exp\left(-\hat{\beta}^{[m]} \cdot \underbrace{y^{(i)} \cdot \hat{h}(\mathbf{x}^{(i)})}_{\{-1,1\}}\right) \\ &= \begin{cases} 0.01 \cdot \exp\left(-0 \cdot 1 \cdot 1\right) & \text{if } i = 1, \ldots, 10 \\ 0.1 \cdot \exp\left(-0 \cdot 1 \cdot -1\right) & \text{if } i = 15, 16, 17, 18 \\ 0.1 \cdot \exp\left(-0 \cdot 1 \cdot 1\right) & \text{if } i = 11, \ldots, 14, 19 \end{cases} \\ &= \begin{cases} 0.01 & \text{if } i = 1, \ldots, 10 \\ 0.1 & \text{if } i = 15, 16, 17, 18 \\ 0.1 & \text{if } i = 11, \ldots, 14, 19 \end{cases} \end{aligned} \tag{6}$$

When the weighted in-sample misclassification rate is 0.5, the weight of the base learner is 0 and we don't do any updates to the weights.

(c) In the third case, all samples with the exception of one with a weight of 0.01 are misclassified. The weighted in-sample misclassification rate can be calculated as:

$$\text{err}^{[m]} = \sum_{i=1}^{n} w^{[m](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}} = 0.99 \tag{7}$$

We can now calculate the weight of the base learner:

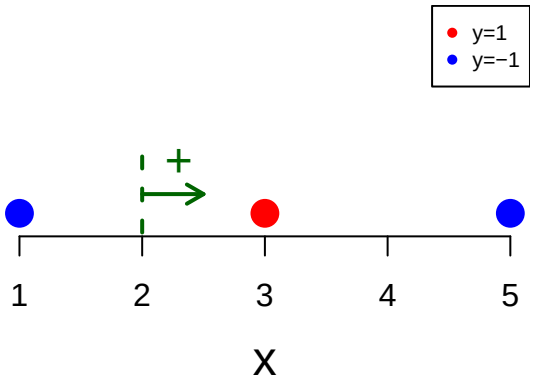
$$\hat{\beta}^{[m]} = \frac{1}{2} \log\left(\frac{1-\text{err}^{[m]}}{\text{err}^{[m]}}\right) = 0.5 \cdot log\left(\frac{0.01}{0.99}\right) = log\left(\sqrt{\frac{1}{99}}\right) \tag{8}$$

The updated weights are then calculated as:

$$\begin{aligned} w^{[m+1](i)} &= w^{[m](i)} \cdot \exp\left(-\hat{\beta}^{[m]} \cdot \underbrace{y^{(i)} \cdot \hat{h}(\mathbf{x}^{(i)})}_{\{-1,1\}}\right) \\ &= \begin{cases} 0.01 \cdot \exp\left(-log\left(\sqrt{\frac{1}{99}}\right) \cdot 1 \cdot -1\right) & \text{if } i = 1, \ldots, 9 \\ 0.01 \cdot \exp\left(-log\left(\sqrt{\frac{1}{99}}\right) \cdot 1 \cdot 1\right) & \text{if } i = 10 \\ 0.1 \cdot \exp\left(-log\left(\sqrt{\frac{1}{99}}\right) \cdot 1 \cdot -1\right) & \text{if } i = 11, \ldots, 19 \end{cases} \\ &= \begin{cases} 0.001 & \text{if } i = 1, \ldots, 9 \\ 0.0995 & \text{if } i = 10 \\ 0.01 & \text{if } i = 11, \ldots, 19 \end{cases} \end{aligned} \tag{9}$$

Solution 2: AdaBoost - Decision Stump

(a) The initial weights for all three points in the dataset is $\frac{1}{3}$. A decision boundary for the first decision stump could be $x = 2$.



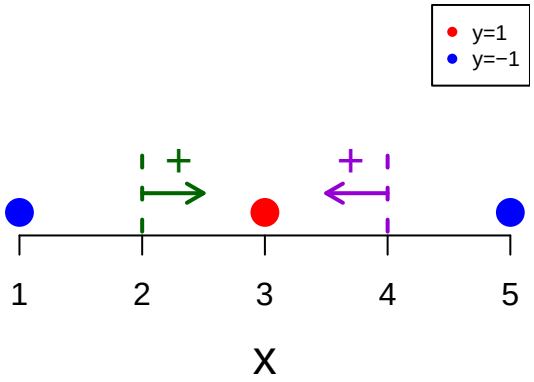
(b) The first stump makes two correct predictions and one incorrect prediction, we can then calculate the weighted in-sample misclassification rate, the weight for the stump and the new data points weights.

$$
\begin{aligned}
\text{err}^{[0]} &= \sum_{i=1}^{n} w^{[0](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[0]}(\mathbf{x}^{(i)})\}} = 0.33 \\
\hat{\beta}^{[0]} &= \frac{1}{2} \log\left(\frac{1-\text{err}^{[0]}}{\text{err}^{[0]}}\right) = 0.5 \cdot log\left(\frac{0.67}{0.33}\right) \approx 0.35 \\
w^{[1](1)} &= w^{[0](1)} \cdot \exp\left(-\hat{\beta}^{[0]} \cdot y^{(1)} \cdot \hat{b}^{[0]}(\mathbf{x}^{(1)})\right) = 0.33 \cdot \exp\left(-0.35 \cdot -1 \cdot -1\right) \approx 0.23 \\
w^{[1](2)} &= w^{[0](2)} \cdot \exp\left(-\hat{\beta}^{[0]} \cdot y^{(2)} \cdot \hat{b}^{[0]}(\mathbf{x}^{(2)})\right) = 0.33 \cdot \exp\left(-0.35 \cdot 1 \cdot 1\right) \approx 0.23 \\
w^{[1](3)} &= w^{[0](3)} \cdot \exp\left(-\hat{\beta}^{[0]} \cdot y^{(3)} \cdot \hat{b}^{[0]}(\mathbf{x}^{(3)})\right) = 0.33 \cdot \exp\left(-0.35 \cdot 1 \cdot -1\right) \approx 0.47
\end{aligned}
\tag{10}
$$

We need to normalize the weights so that they sum up to one:

$$
\begin{aligned}
w^{[1](1)} &= \frac{w^{[1](1)}}{\sum_{i=1}^{n} w^{[1](i)}} = \frac{0.23}{0.23+0.23+0.47} \approx 0.25 \\
w^{[1](2)} &= \frac{w^{[1](2)}}{\sum_{i=1}^{n} w^{[1](i)}} = \frac{0.23}{0.23+0.23+0.47} \approx 0.25 \\
w^{[1](3)} &= \frac{w^{[1](3)}}{\sum_{i=1}^{n} w^{[1](i)}} = \frac{0.47}{0.23+0.23+0.47} \approx 0.50
\end{aligned}
\tag{11}
$$

(c) As the training error is not yet 0, we do a second stump using the new weights, The decision boundary is $x = 4$. The stump makes two correct predictions and one incorrect prediction:



We calculate the weighted in-sample misclassification rate and the weight for the stump.

$$\begin{aligned}\text{err}^{[1]} &= \sum_{i=1}^{n} w^{[1](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[1]}(\mathbf{x}^{(i)})\}} = 0.25 \\ \hat{\beta}^{[1]} &= \frac{1}{2}\log\left(\frac{1-\text{err}^{[1]}}{\text{err}^{[1]}}\right) = 0.5 \cdot log\left(\frac{0.75}{0.25}\right) \approx 0.54\end{aligned} \tag{12}$$

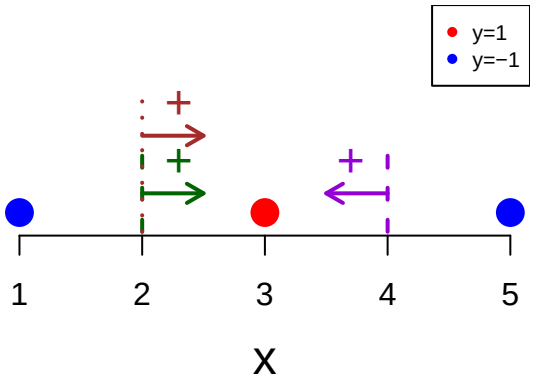
We can see that for the left-most point and the right-most point, there is a disagreement between our two stumps. In the case of the right-most point, the second stump has a bigger weight than the first one, so we will classify it correctly as $y = -1$. Unfortunately, in the case of the left-most point, we will classify it incorrectly as $y = 1$. Let's calculate the new weights:

$$\begin{aligned}w^{[2](1)} &= w^{[1](1)} \cdot \exp\left(-\hat{\beta}^{[0]} \cdot y^{(1)} \cdot \hat{b}^{[1]}(\mathbf{x}^{(1)})\right) = 0.25 \cdot \exp\left(-0.54 \cdot 1 \cdot -1\right) \approx 0.42 \\ w^{[2](2)} &= w^{[1](2)} \cdot \exp\left(-\hat{\beta}^{[0]} \cdot y^{(2)} \cdot \hat{b}^{[1]}(\mathbf{x}^{(2)})\right) = 0.25 \cdot \exp\left(-0.54 \cdot 1 \cdot 1\right) \approx 0.15 \\ w^{[2](3)} &= w^{[1](3)} \cdot \exp\left(-\hat{\beta}^{[0]} \cdot y^{(3)} \cdot \hat{b}^{[1]}(\mathbf{x}^{(3)})\right) = 0.5 \cdot \exp\left(-0.54 \cdot -1 \cdot -1\right) \approx 0.29\end{aligned} \tag{13}$$

We need to normalize the weights so that they sum up to one:

$$\begin{aligned}w^{[2](1)} &= \frac{w^{[2](1)}}{\sum_{i=1}^{n} w^{[2](i)}} = \frac{0.42}{0.42+0.15+0.29} \approx 0.5 \\ w^{[2](2)} &= \frac{w^{[2](2)}}{\sum_{i=1}^{n} w^{[2](i)}} = \frac{0.23}{0.42+0.15+0.29} \approx 0.17 \\ w^{[2](3)} &= \frac{w^{[2](3)}}{\sum_{i=1}^{n} w^{[2](i)}} = \frac{0.47}{0.42+0.15+0.29} \approx 0.33\end{aligned} \tag{14}$$

We will start now with the third iteration. We will add a new stump on $x = 2$.



We calculate the weighted in-sample misclassification error for this stump:

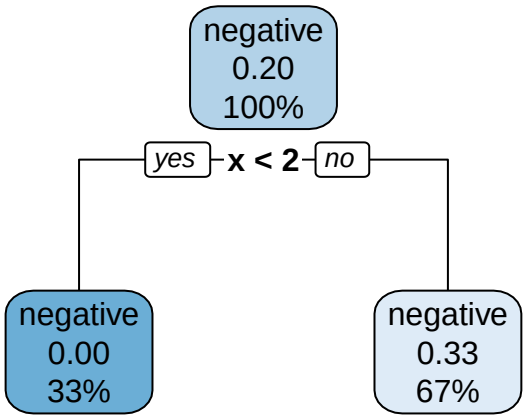
$$\begin{aligned} \text{err}^{[2]} &= \sum_{i=1}^{n} w^{[2](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[2]}(\mathbf{x}^{(i)})\}} = 0.33 \\ \hat{\beta}^{[2]} &= \frac{1}{2} \log\left(\frac{1-\text{err}^{[2]}}{\text{err}^{[2]}}\right) = 0.5 \cdot log\left(\frac{0.67}{0.33}\right) \approx 0.35 \end{aligned} \tag{15}$$

We can see that we are going back to a similar case as the first iteration, it seems like we will keep creating new stumps on $x = 2$ and $x = 4$ and the model will never reach a training error of zero. Let's implement Adaboost with stumps in R to find out:

```
library(rpart)
library(rpart.plot)
# Define x
x <- c(1, 3, 5)
# Define y
y = as.factor(c('negative', 'positive', 'negative'))
df = data.frame(x, y)

w <- rep(1 / 3, 3)
M <- 100
err <- rep(0,M)
beta <- rep(1, M)
b_list <- list()
f <- rep(0,3)
m <- 1
while (m < M) {
  cat("m =", m, ":\n")
  # 3: Fit classifier with weights
  tree = rpart(y ~ x,
               data = df,
               weights = w,
               control = rpart.control(
                 minsplit = 0,
                 maxdepth = 1,
                 cp = -1
               ))
  rpart.plot(tree,main = paste("Stump for iteration: ",m))
  cat("split point:", tree$splits[,"index"], "\n")
  # 4: Calculate error
  b <- predict(tree, df, type="class")
  indicator <- b!=y
  err[m] <- sum(w*indicator)
  cat("error unweighted:", sum(indicator)/3, "\n")
  cat("error weighted:", err[m], "\n")

  # 5: Calculate beta
  beta[m] <- 0.5 * log((1-err[m])/err[m])

  # Update weigths
  y_mal_b <- -(indicator*2-1)
  w_unnormalized <- w * exp (-beta[m]*y_mal_b)
  w_normalized <- w_unnormalized/sum(w_unnormalized)
  w <- w_normalized
  cat("new weights after this iteration:", w, "\n")
  b_list[[m]] <- b
  f <- f + beta[m]*(as.numeric(b)*2-3)
  cat("f:", f, "\n")
  h <- f>0
  err_ens <- sum(h!=(y=="positive"))/3
```

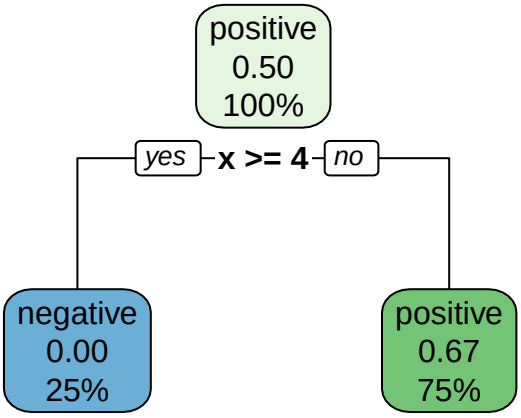
```
  cat("error of ensemble:", err_ens, "\n \n")
  if(err_ens==0){
    cat( "We reached 0% error rate on the training dataset!")
    m=M
  }else{
    m <- m+1
  }
}

## m = 1 :
```

Stump for iteration: 1



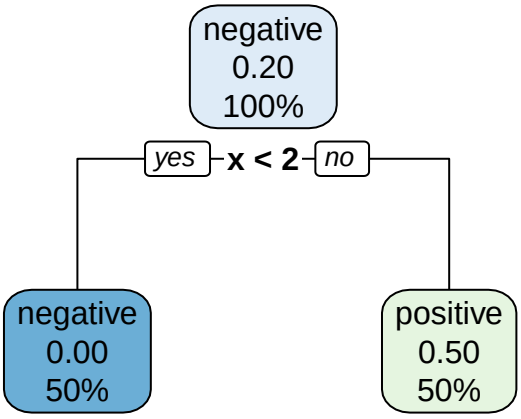

```
## split point: 2
## error unweighted: 0.3333333
## error weighted: 0.3333333
## new weights after this iteration: 0.25 0.5 0.25
## f: -0.3465736 -0.3465736 -0.3465736
## error of ensemble: 0.3333333
##
## m = 2 :
```

Stump for iteration: 2



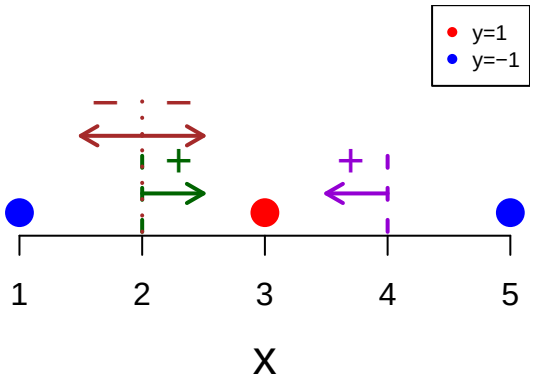

```
## split point: 4
## error unweighted: 0.3333333
## error weighted: 0.25
## new weights after this iteration: 0.5 0.3333333 0.1666667
## f: 0.2027326 0.2027326 -0.8958797
## error of ensemble: 0.3333333
##
## m = 3 :
```

Stump for iteration: 3



```
## split point: 2
## error unweighted: 0.3333333
## error weighted: 0.1666667
## new weights after this iteration: 0.3 0.2 0.5
## f: -0.6019864 1.007452 -0.09116078
## error of ensemble: 0
##
## We reached 0% error rate on the training dataset!
```

The code shows that we reached convergence in only 3 steps. Why our results are not the same? The reason behind the difference is that rpart allows stumps where all the leaves lead to the same group, as in the 1st iteration. If we take this into account, we could build again our third stump:



The weighted in-sample misclassification error and voting power of the stump are calculated:

$$\begin{aligned} \text{err}^{[2]} &= \sum_{i=1}^{n} w^{[2](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[2]}(\mathbf{x}^{(i)})\}} = 0.17 \\ \hat{\beta}^{[2]} &= \frac{1}{2} \log\left(\frac{1-\text{err}^{[2]}}{\text{err}^{[2]}}\right) = 0.5 \cdot log\left(\frac{0.83}{0.17}\right) \approx 0.79 \end{aligned} \tag{16}$$

Let's check how the points are classified:

$$\begin{aligned} \hat{y_1} &= \text{sign}\left(\sum_{i=1}^{m} \hat{\beta}^{[i]} \hat{b}^{[i]}(\mathbf{x}^{(1)})\right) = \text{sign}(-0.35 + 0.54 - 0.79) = -1 \\ \hat{y_2} &= \text{sign}\left(\sum_{i=1}^{m} \hat{\beta}^{[i]} \hat{b}^{[i]}(\mathbf{x}^{(2)})\right) = \text{sign}(+0.35 + 0.54 - 0.79) = 1 \\ \hat{y_3} &= \text{sign}\left(\sum_{i=1}^{m} \hat{\beta}^{[i]} \hat{b}^{[i]}(\mathbf{x}^{(3)})\right) = \text{sign}(+0.35 - 0.54 - 0.79) = -1 \end{aligned} \tag{17}$$

In this case, we reached zero error in training in the third iteration.